İSTANBUL SAĞLIK TEKNOLOJİ ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR/YAZILIM/MEKATRONİK MÜHENDİSLİĞİ

Mikroişlemciler ve Gömülü Sistemler

Gömülü Sistem Yazılımı/Donanımı ve Endüstriyel Uygulamaları

Yrd. Doç. Dr. Tuncay UZUN

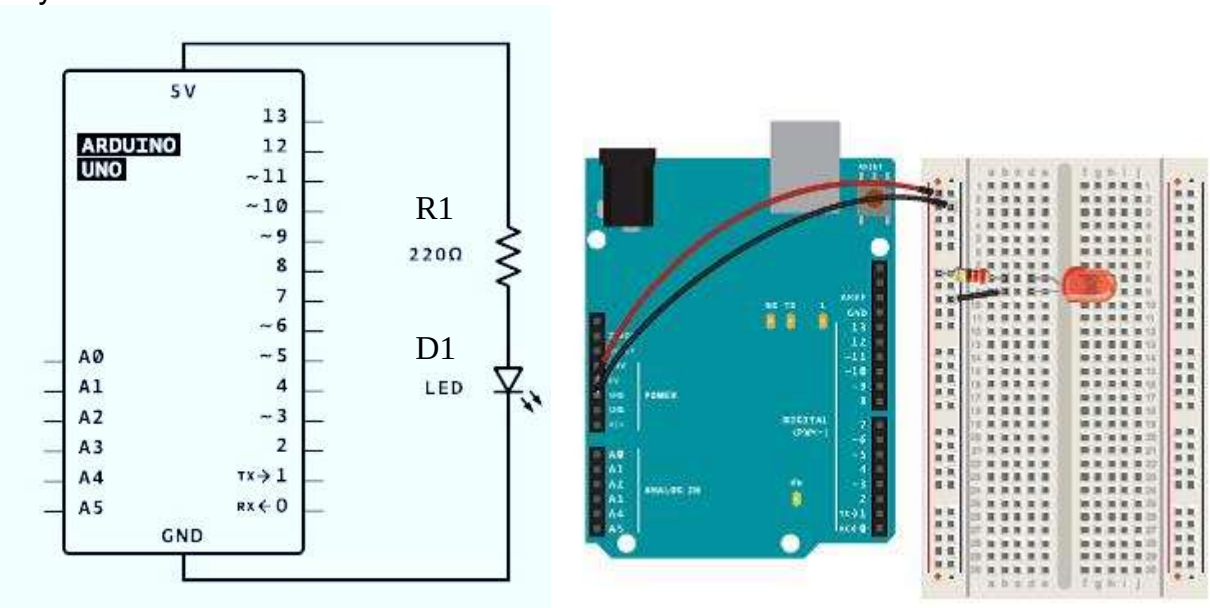
MİKROİŞLEMCİLER ve GÖMÜLÜ SİSTEMLER UYGULAMA PROGRAMI				
Hafta	**Uygulama Konusu**	**Tarih**	**Saat**	**Grup**
1	DERS	13 Şubat		
2	DERS	20 Şubat		
3	Uygulama Donanımı ve Yazılımı Tanıtım/Giriş	27 Şubat	**10:30**	**1**
			11:30	**2**
4	1.a) LED Görsel Uygulama-1 (LED Yakma) 1.b) LED Görsel Uygulama-2 (Buton ile LED Yakma)	5 Mart	**10:30**	**1**
			11:30	**2**
5	2.a) Analog Veri Okuma ve Seri Haberlesme 2.b) Potansiyometre ile LED Yakma 2.c) Kara şimşek Uygulaması	12 Mart	**10:30**	**1**
			11:30	**2**
6	3.a) Algılayıcı (Sensor) Uygulama-1 (Otomatik Lamba) 3.b) Çok Renkli (RGB) LED Uygulaması 3.c) Algılayıcı (Sensor) Uygulama-2 (Sıcaklık Ölçümü)	19 Mart	**10:30**	**1**
			11:30	**2**
7	4. Ultrasonik Sensor ile Park Sensörü Uygulaması	26 Mart	**10:30**	**1**
			11:30	**2**
8	**ARA SINAV**			
9	Resmi Tatil (Ramazan Bayramı)	9 Nisan		
10	5.a) LED Gösterge Uygulaması 5.b) LCD Gösterge Uygulaması	16 Nisan	**10:30**	**1**
			11:30	**2**
11	Resmi Tatil (Ulusal Egemenlik ve Çocuk Bayramı)	23 Nisan		
12	Proje (2 hafta)	30 Nisan	**10:30**	**1**
13	Proje (2 hafta)	7 Mayıs	**10:30**	**2**
14	Proje Teslim	14 Mayıs	**10:30**	**1**
15	Proje Teslim	21 Mayıs	**10:30**	**2**

Deney 1: LED Görsel Uygulamalar

Deneyin Amacı:
Eğitim sistemini ile deney tablasını bağlantısını yapmayı, uygulama devresini kurmayı, giriş/çıkış portlarının kullanılmasını öğrenmek için tümleşik devrelerinin çalışma yöntemlerinin, donanımlarının incelenmesi ve programlanmasının yapılmasıdır.

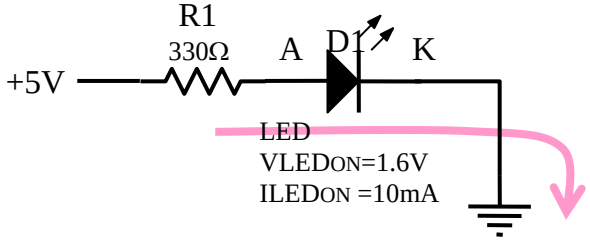
Deney Öncesi Yapılacak İşlemler:
Temel devre yasalarını: "Kirchhoff" Akım/Gerilim ve "Ohm" yasasının öğrenim tekrarı yapılacaktır. Bu deneyde kullanılan elektronik devre malzemelerinin işlevleri, kullanım özellikleri, malzemenin fiziksel görünümü, adının ve değerinin üzerindeki verilerden okunarak elde edilmesi öğrenilecektir. Burada verilen uygulama programlarını inceleyerek her komutun açıklamasını yanına kısaca yazınız.

Deneyde Yapılacak İşlemler:
1. LED Görsel Uygulama-1 (LED Yakma).
a) Bu uygulamada eğitim kartının güç bağlantı uçları kullanılarak bir ışık yayan diyotun (LED) ışık vermesini sağlayan aşağıda verilen elektrik devresi kurulacaktır. Bu devre çalıştırılarak akım ve gerilim değerleri incelenecek sonuçlar yorumlanacaktır.



Şekil 1. (a) Elektrik Devresi (b) Montaj Görüntüsü

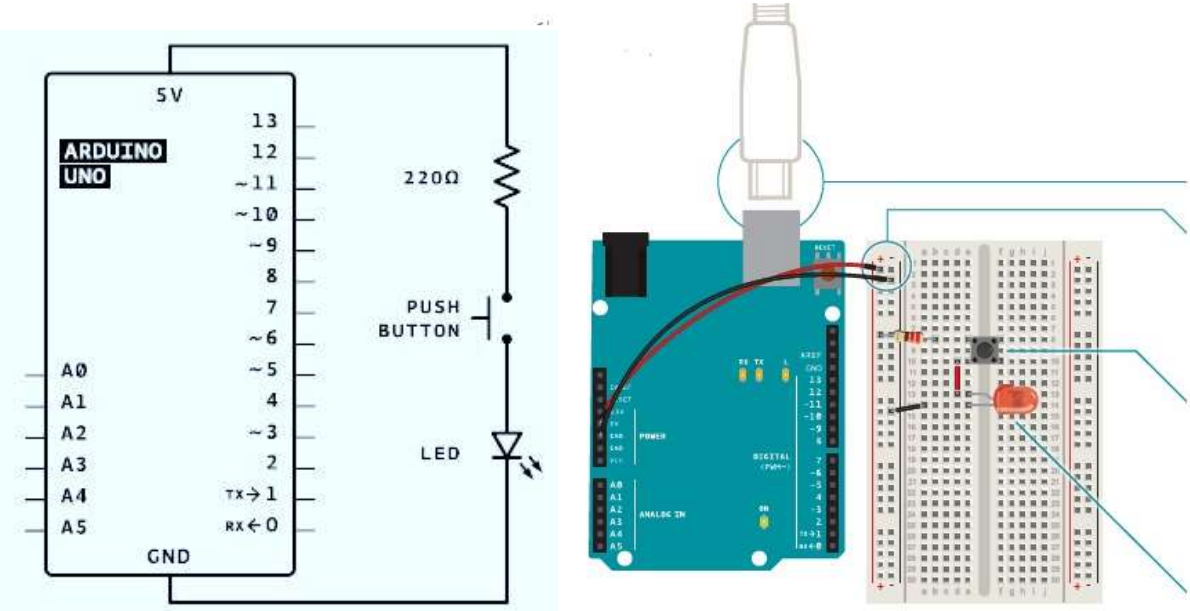
$$R1 = \frac{VCC - VLEDON}{ILEDON} = \frac{5V - 1{,}6V}{10mA} = 340\Omega$$ *En yakın standart direnç* R1=330Ω



Şekil 2. LED'in pozitif lojik ile çalıştırılması için eşdeğer devre

2. LED Görsel Uygulama-2 (Buton ile LED Yakma).

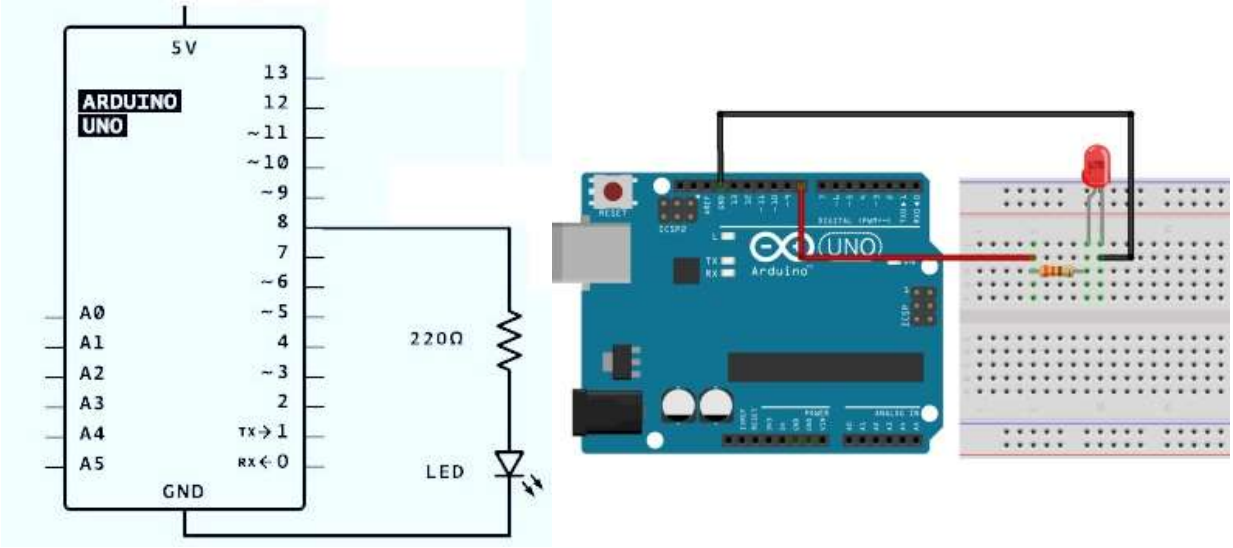
b) Bu uygulamada eğitim kartının güç bağlantı uçları kullanılarak bir ışık yayan diyotun (LED) ışık vermesini, diyota seri bağlı bir anahtar ile kontrol edilmesini sağlayan aşağıda verilen elektrik devresi kurulacaktır. Bu devre çalıştırılarak akım ve gerilim değerleri incelenecek sonuçlar yorumlanacaktır.



Şekil 3. (a) Elektrik Devresi (b) Montaj Görüntüsü

3. LED Görsel Uygulama-3 (Buton ile LED'in Kontrol edilmesi).

c) Bu uygulamada eğitim kartının Giriş/Çıkış uçları kullanılarak bir ışık yayan diyotun (LED) ışık vermesini sağlayan aşağıda verilen elektrik devresi kurulacaktır. Ayrıca aşağıda verilen program yazılarak eğitim kartına yüklenecektir. Bu program ile devre çalıştırılarak incelenecek sonuçlar yorumlanacaktır.



Şekil 4. (a) Elektrik Devresi (b) Montaj Görüntüsü

$$R1 = \frac{VOH - VLED_{ON}}{ILED_{ON}} = \frac{4V - 1{,}6V}{10mA} = 240\Omega$$

En yakın standart direnç R1=220Ω

Uygulama programı:

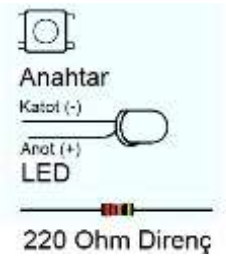
```
/*
Bu uygulama bir LED'i tekrar tekrar bir saniye süreyle yakar, ardından bir saniye süreyle söndürür.
*/
// setup fonksiyonu, reset tuşuna bastığınızda veya karta güç verdiğinizde bir kez çalışır!
void setup() {
  // 8 nolu sayısal uç çıkış olarak koşullandırılır
  pinMode(8, OUTPUT);
}

// loop fonksiyonu ana programı sonsuza kadar tekrar tekrar çalışır
void loop() {
  digitalWrite(8, HIGH);   //LED yanık, ışık verir(lojik “1”)
  delay(1000);               // bir saniye bekle
  digitalWrite(8, LOW);  //LED sönük, ışık vermez(lojik “0”)
  delay(1000);               // bir saniye bekle
}
```

Sorular:

1. Eğitim kartının G/Ç uçlarının çalışma yöntemleri ve özellikleri nelerdir?
2. Uygulama programı çalışırken hangi uçtan hangi işaret gönderilir?
3. Uygulama programının başlangıcında bulunan “setup” fonksiyonu ne için kullanılır?
4. Uygulama programının son kısmında bulunan “loop” fonksiyonu ne için kullanılır?
5. “pinMode” komutunun özelliği ve çalışma şekli nasıldır?
6. “digitalWrite” komutunun özelliği ve çalışma şekli nasıldır?
7. “delay” zamanlama komutunun özelliği ve çalışma şekli nasıldır?
8. Uygulama programı çalıştırıldığında LED nasıl görünür?

Malzeme Listesi



1. Anahtar
2. Işık Yayan Diyot (**L**ight **E**mitting **D**iode, LED)
3. 220 Ohm Direnç (Kırmızı Kırmızı Kahverengi)

04

Arduino ile Analog Okuma ve Seri Haberleşme

robotistan BLOG

Uygulamanın blog yazısına aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinodersleri

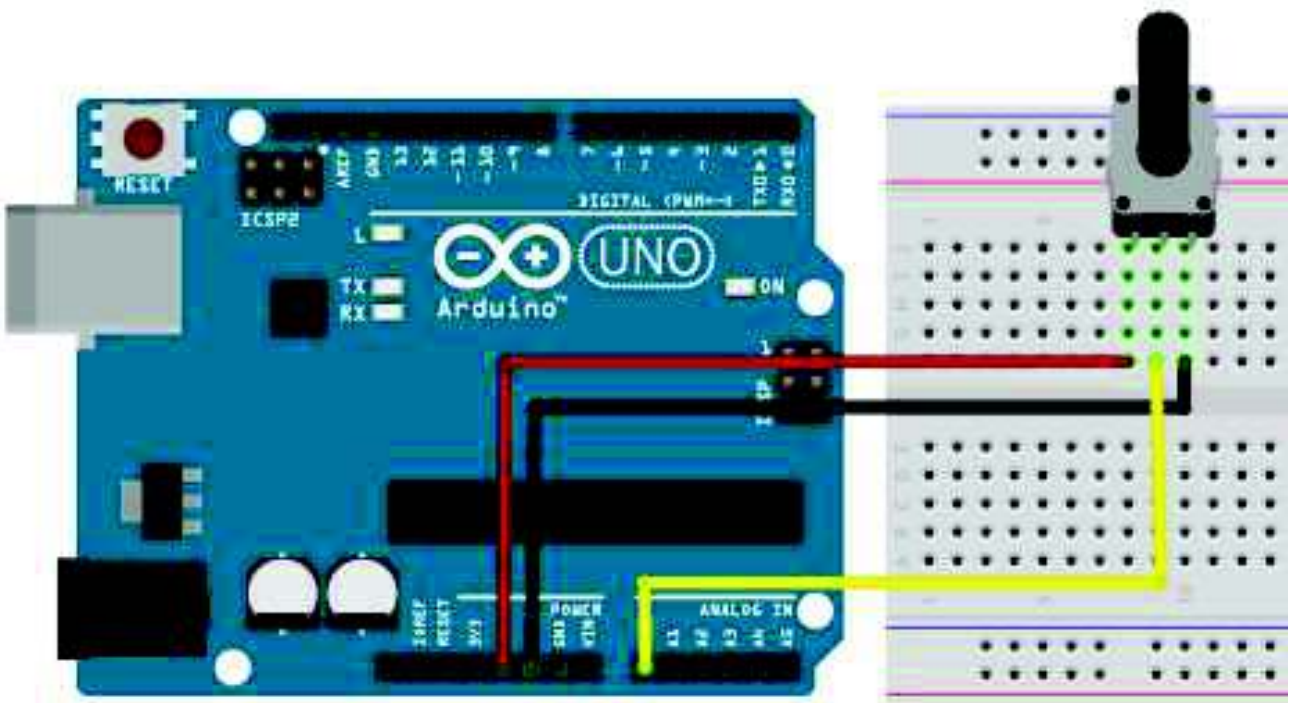
YouTube

Uygulamanın videosuna aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinovideodersler

Gerekli malzemeler:

- Arduino Uno
- Breadboard
- 10k Ohm Potansiyometre
- 3 Adet Erkek-Erkek Jumper Kablo

Arduino kartının üzerine baktığınızda "Analog Input" pinlerini göreceksiniz. Bu pinleri kullanarak dijitalden analog sinyale dönüşüm yaparak bu pindeki voltajı okumamız mümkün. Arduino dijital okuma yaparken 0V (sıfır) ve 5V okuyabilmektedir. Bu iki uç değer arasında ara değerler gelirse bunu algılayamamakta ve gelen voltajı eşik değerine göre 0V veya 5V olarak kabul etmektedir. Analog pinler sayesinde 0V'dan 5V'ta kadar ara gerilim değerlerini de algılayarak dijitale çevirebiliyoruz. Ara değerlerdeki sinyalleri elde edebilmek için ayarlı direnç (potansiyometre) kullanacağız. Uygulamamızda analog giriş pininden gelen gerilimin sayısal karışılığını seri porttan okuma işlemini sağlayacağız. Devremizi kurup kod kısmına geçelim.



```
#define potpin A0

int deger=0;

void setup() {
  Serial.begin(9600);
  Serial.println("Pot Değer Okuma");
}

void loop() {
  deger = analogRead(potpin);
  Serial.println(deger);
  delay(300);
}
```

Önceki kodlarımızda da yaptığımız gibi define işlemi ile "A0" pinine "potpin" ismini veriyoruz. Sonraki satırda analog pinden okuduğumuz değerleri saklamak için "integer" türünde ve değer isminde değişken tanımlıyoruz.

"setup" kısmında önceki yazılımlarımızda dijital giriş-çıkış kullandığımızdan dolayı, pinleri ne olarak kullanacağımıza göre ayarlıyorduk. Analog okuma yaparken giriş çıkış tanımlamamıza gerek yok bu sebepten dolayı bu yazılımda "pinMode" komutunu kullanmıyoruz.
Okuduğumuz verileri bilgisayara göndermek için seri haberleşme başlatmamız gerekiyor. Bu seri haberleşme sayesinde Arduino ile bilgisayar USB bağlantı üzerinden haberleşecek ve istediğimiz verileri bilgisayara aktarabileceğiz.

"Serial.begin(9600);" satırı ile bu haberleşmeyi başlatıyoruz. Arduino kodu çalıştırmaya başladığında ilk olarak bilgisayar ile haberleşmeyi başlatacaktır. Haberleşme başladıktan sonra "Serial.println("Pot Deger Okuma");" satırı ile "Pot Deger Okuma" yazısı bilgisayarda seri monitöre yazdırılacaktır. "Serial.print " ve "serial.println" komutlarını aşağıda detaylı açıklayacağız.

05

Potansiyometre ile LED Yakma

robotistan BLOG

Uygulamanın blog yazısına aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinodersleri

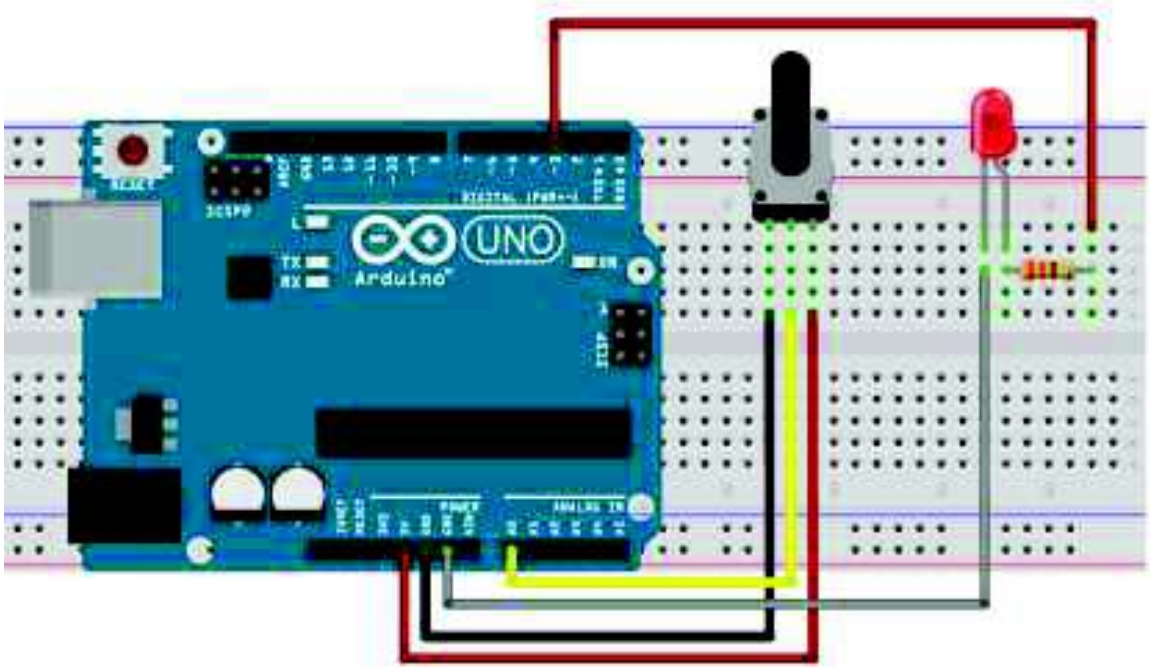
YouTube

Uygulamanın videosuna aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinovideodersler

Gerekli malzemeler:

- Arduino UNO
- Breadboard
- 10k Ohm potansiyometre
- Kırmızı LED
- 330 Ohm Direnç (Turuncu - Turuncu - Kahverengi)
- 5 Adet Erkek-Erkek Jumper Kablo

Önceki uygulamamızda analog pinden gerilim değerini okumuştuk bu uygulamamızda yine analog pinden gelen değere göre LED'in parlaklığını kontrol edeceğiz. İlk led yakma uygulamamızda dijital çıkış kullandığımızdan dolayı LED'e 0V veya 5V gönderebiliyorduk. Bu yüzden led ya sönüyordu yada yanıyordu. Arduino'nun yeni bir özeliğini kullanarak LED'e 0-5V aralığında ara değerlerde gerilimde gönderebileceğiz. Bu gerilim kontrolü sayesinde LED'in parlaklığını ayarlayabileceğiz. Bu uygulamaya kadar dijital giriş-çıkış ve analog girişi öğrendik. Bu uygulamayla birlikte analog çıkış yani PWM özelliğini öğreneceğiz. Şimdi devremizi kurup hemen kod kısmına geçelim.



PWM (Pulse with Modulation) , sinyal genişlik modülasyonun kısaltmasıdır.Bu özellik Arduino Uno üzerine 6 pinde mevcut yaklaşık işareti (~) bulunan pinlerden (3,5,6,9,10 ve 11. Pinler) PWM ile ilgili detaylı bilgiler için uygulama sonundaki kare kodu taratarak bu uygulamaya ait olan videoyu izleyebilirsiniz. Devremizi kurduktan sonra hemen kod kısmına geçelim.

Arduino

```
#define led 3
#define pot A0

void setup() {
}

void loop() {
  int deger = analogRead(pot);
  deger = map(deger,0,1023,0,255);
  analogWrite(led,deger);
}
```

Bu uygulamamız da dijital giriş-çıkış kullanmadığımızdan dolayı yine "setup" kısmında bir ayarlama yapmıyoruz.

Ana program döngümüzde POT'tan veriyi okuyup bu veriyi LED'e göndermek istiyoruz. İlk olarak "analogRead" komutu ile potansiyometreden veriyi okuyoruz. Okuduğumuz değeri "deger" değişkenine yazıyoruz. İkinci satırda ise "map" komutunu kullanarak 0 ile 1023 arasında gelen değeri 0 ile 255 arasına oranlıyoruz.

Analog okumayı 10 bit (2^10 = 1024) çözünürlükte yaparken, analog yazmayı 8 bit (2^8 = 256) çözünürlükte yapabiliyoruz. Okuduğumuz veriyi, çıkışa göre oranlayıp yazdırmamız gerekiyor. "map" komutu yerine dilerseniz direk olarak 4'e de bölebilirsiniz. Oranlama işleminden sonra "analogWrite" komutu ile PWM pinlerinden çıkış verebiliriz.

06

Arduino ile Karaşimşek Uygulaması

robotistan BLOG

Uygulamanın blog yazısına aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinodersleri

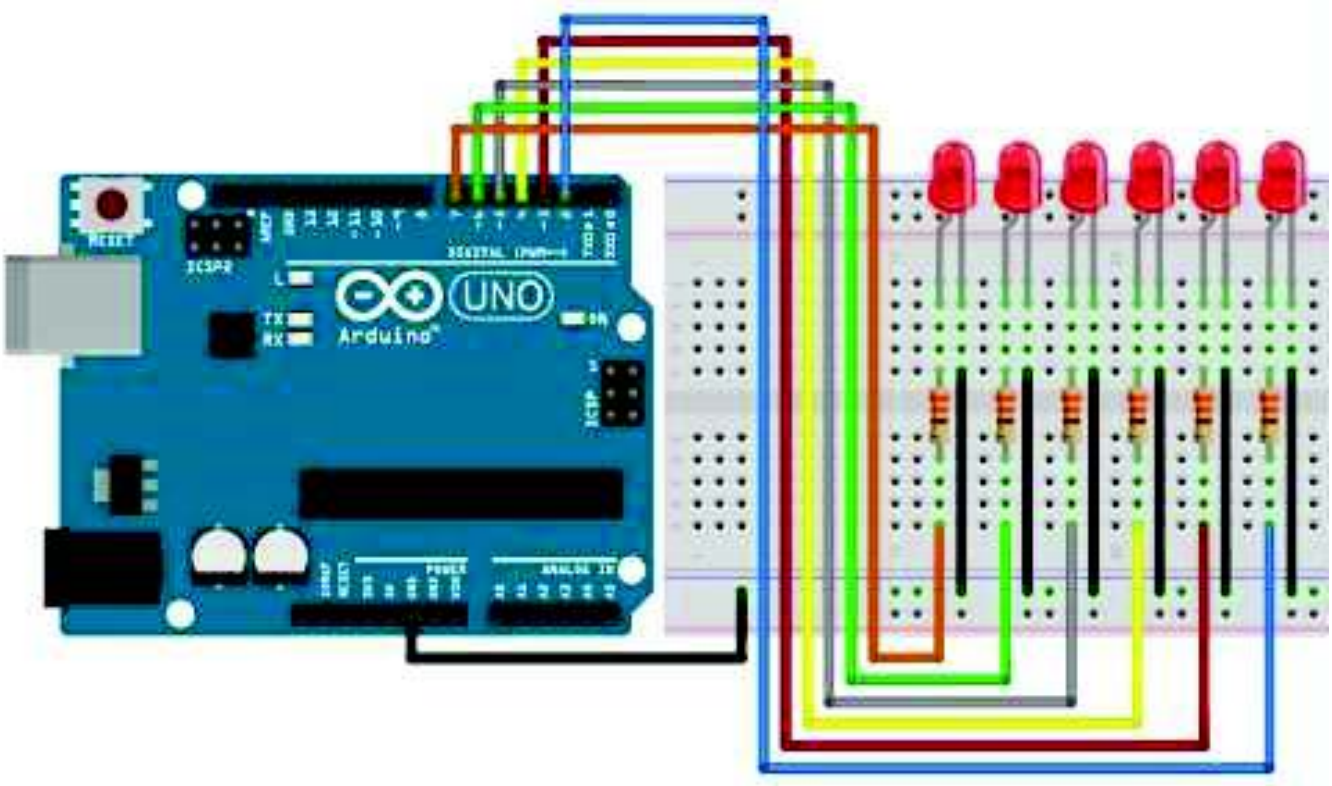
YouTube

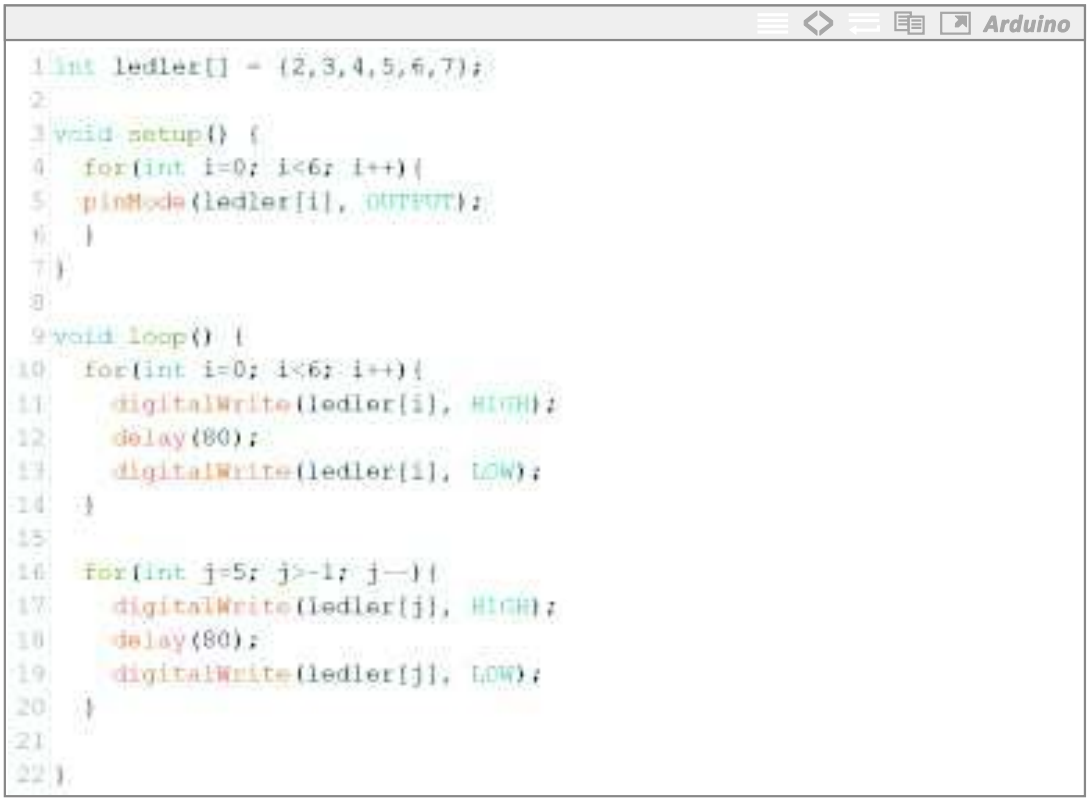
Uygulamanın videosuna aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinovideodersler

Gerekli malzemeler:

- Arduino Uno
- Breadboard
- 13 Adet Erkek-Erkek Jumper kablo
- 6 Adet LED
- 6 Adet 330 Ohm Direnç (Turuncu - Turuncu - Kahverengi)

Bu uygulamamızda "for" döngüsünü kullanımını göreceğiz. "for" döngüsü ardı sıra yapılması gereken işlemlerde kullanılabilir. Uygulamamızda 6 adet LED'i yakmak için hepsini çıkış vermemiz ve sırasıyla yakmamız gerekecek. Bunu normal öğrendiğimiz çıkış tanımlama ve LED yakma söndürme komutları ile rahatlıkla uygulayabilmekteyiz. "for" döngüsü kullanmadan yazılan kodda değişiklik yapmak istediğinizde her satırda tekrar tekrar değişiklik yapmanız gerekecek, "for" döngüsünde ise hem kodu anlamak hem yazmak hem de değişiklik yapmak istediğiniz de çok daha hızlı şekilde ilerleyebileceksiniz. Devremizi kurduktan sonra hemen kod kısmına geçelim.





Dijital pinleri tek tek tanımlarken "int" veya "#define" ile çıkışlara isim tanımlaması yapıyorduk. Bu sefer 6 çıkış birden kullacağımızdan dolayı dizi kullanarak ilerleyeceğiz. Dizileri, değişkenleri barındıran bir küme olarak düşünebilirsiniz. Kodun üst kısmında içerisindeki değişken türleri "int" (tamsayı) olan ve ismi "ledler" olan dizi tanımlıyoruz. Dizi elemanlarını ise içerisine virgüller ile ayırarak yazıyoruz. Dijital çıkışlardan 2 numaranda 7 numaraya kadar kullanacağımız için elamanlarımızı bu şekilde belirledik.

Dizinin elemanlarını çoğaltadabilirsiniz. Dizi tanımlama yaparken eğer istersek "ledler[]" ifadesi içerisine dizinin kaç elemanlı olacağını yazabiliriz. Örneğin "int ledler[6] = {2,3,4,5,6,7};" gibi
Diziden eleman çağırırken ilk elemanın numarası 0'dan (sıfırdan) başlar. İstediğiniz elemanı çağırmak içinde "setup" veya "loop" içerisinde "ledler[*dizi elemanı sıra numarası*]" ifadesini kullanabilirsiniz. Yani eğer sıfırıncı dizi elemanını çağırmak için "ledler[0]" yazarsanız bu 2'ye eşit olacaktır. 5. Dizi elemanını çağırmak için "ledler[5]" yazarsanız buda 7'ye eşit olacaktır.

Uygulamamızda 6 adet LED'i yakmak istiyoruz bu durumda 6 adet dijital pinlerden çıkış belirlememiz gerekiyor. "for" döngüsünün parantez içerisinde ilk noktalı virgüle kadarki kısım döngü için kullanılacak koşulun değişkeni olarak tanımlanıyor.

Bu yazılımda sadece "for" döngüsü için kullanılacağından dolayı parantez içerisinde tanımladık. İsterseniz hali hazırda farklı bir değişkeninizi veya değişkeni yazılımın üstünde tanımlayıp sonradan burada kullanabilirsiniz.

"İ<6" ifadesi ise "for" döngüsünün koşulunu belirliyor. "i" değişkeni 6'dan küçük olduğu sürece "for" döngüsü içerisindeki kod satırlarını tekrarlayacak. Eğer "i" değişkeni 6'ya eşit veya büyük olursa artır "for" döngüsüne yapmayıp döngünün bittiği yerden kodu işletmeye devam edecek.

"i++" ifadesi ile "for" döngüsünü her yaptığımızda "i" değişkeninin değerini 1 arttırmasını istiyoruz. Böylelikle "i" değişkeninin ilk değeri 0(sıfır) oluyor. Dijital çıkış verdiğimiz komutlar içerisinde de "i" değişkenini yerine yazdığınızda diziden elemanı çağırıyor. Yani "pinMode(ledler[0], OUTPUT)" yazdığınızda yazılım "ledler" dizisinden sıfırıncı elemanı çağırarak "pinMode(2, OUTPUT)" olarak algılıyor. Bu sayede 2. pini çıkış olarak tanımlıyoruz. "for" döngüsü 0'dan 5'e kadar bunu yapacağı için 6 adet pini çıkış olarak tek satır ile tanımlamış oluyoruz.

"loop" kısmında da "setup" kısmındaki gibi "for" döngüsü kullanımı aynı mantıkla ilerliyor. Bu seferde çıkış tanımlamak yerine "digitalWrite" komutu ile sırasıyla 6 adet LED'i yakıp söndürüyoruz. "for" döngüsü ile ilgili detaylı video anlatımına aşağıdaki kare kodu taratarak izleyebilirsiniz.

07

LDR ile Otomatik Lamba Uygulaması

robotistan BLOG

Uygulamanın blog yazısına aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinodersleri

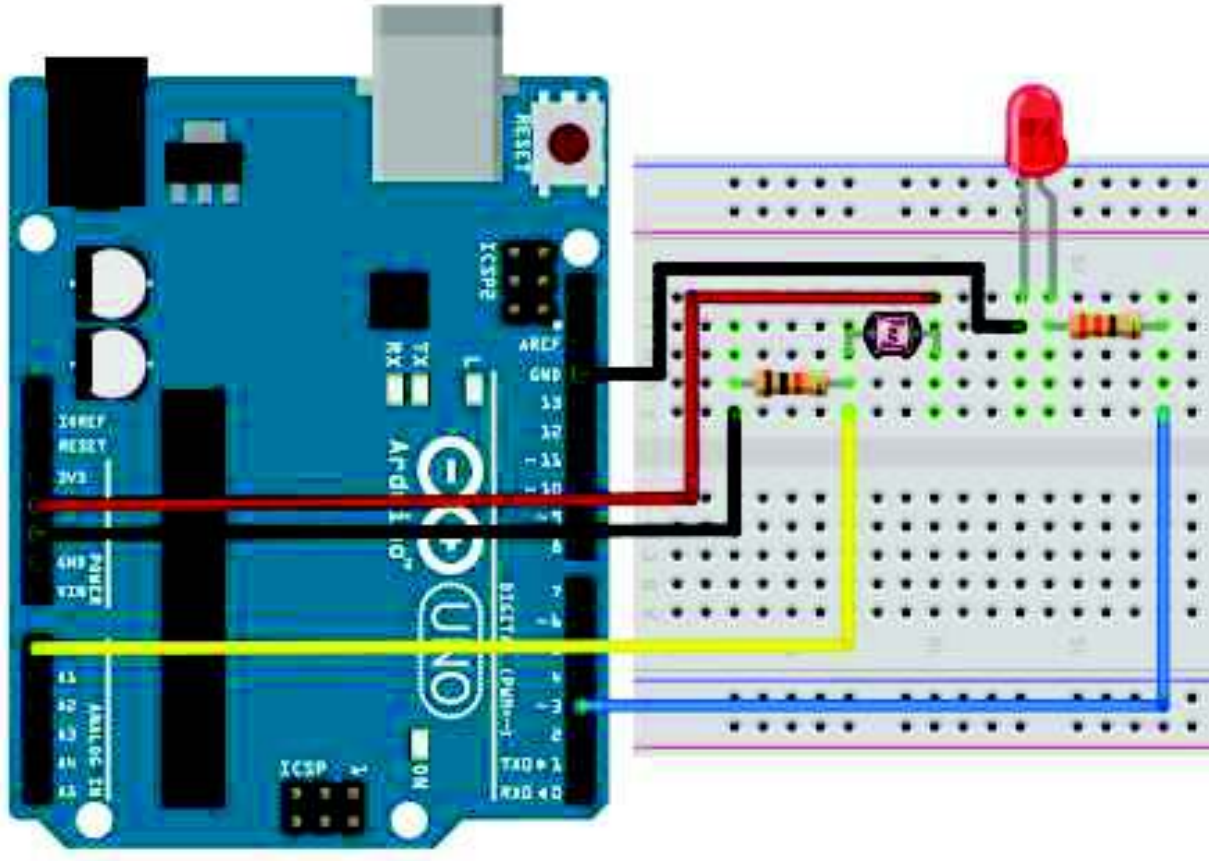
YouTube

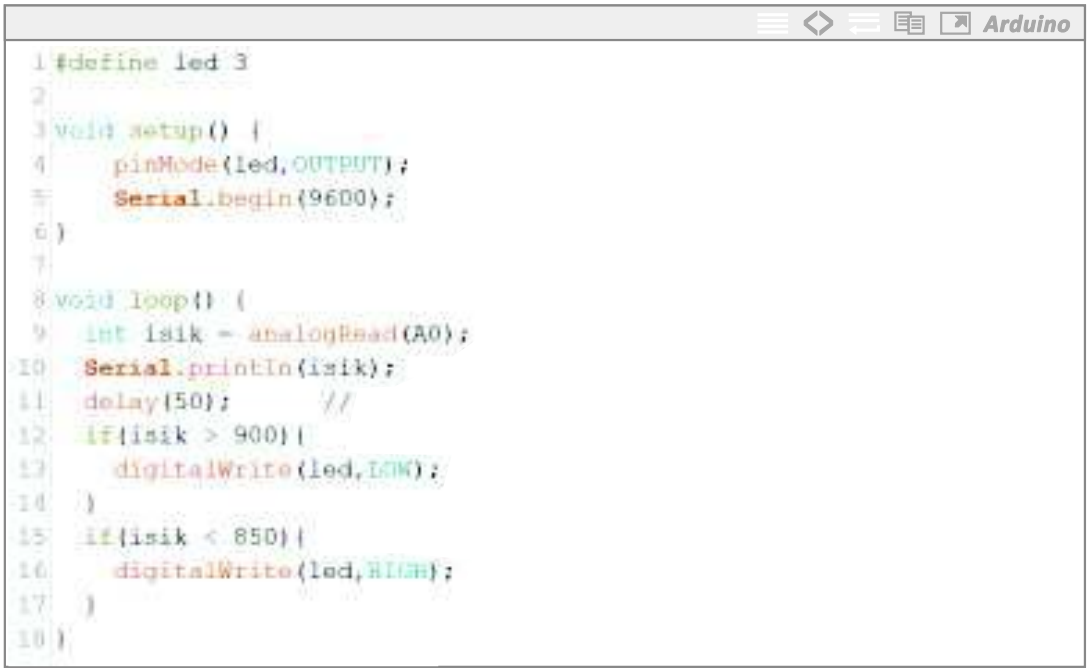
Uygulamanın videosuna aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinovideodersler

Gerekli malzemeler:

- Arduino Uno
- Breadboard
- 5 Adet Erkek-Erkek Jumper Kablo
- 330 Ohm Direnç (Turuncu-Turuncu-Kahverengi)
- 10K Ohm Direnç (Kahverengi-Siyah-Turuncu)
- 5mm Kırmızı LED
- 5mm LDR

Bu uygulamamızda ortamdaki ışığı algılayabilen LDR'den veri okuyup, bu veriye göre LED'imizi yakıp söndüreceğiz. LDR (Light Dependent Resistance) yani fotodirenç ortamdaki ışık miktarına göre direncini değiştirir. Bu direnç değişimini Arduino kartı ile algılayabiliriz.Bu sayede ortamdaki ışık miktarını bilebildiğimiz için ortam karanlık olduğunda LED'i yakıp, aydınlık olduğunda LED'i söndürerek otomatik bir lamba yapacağız. Aynı zamanda aldığımız verileri bilgisayara gönderip, serial monitör üzerinde de görüntüleyeceğiz. Hemen devremizi kurarak başlayalım.



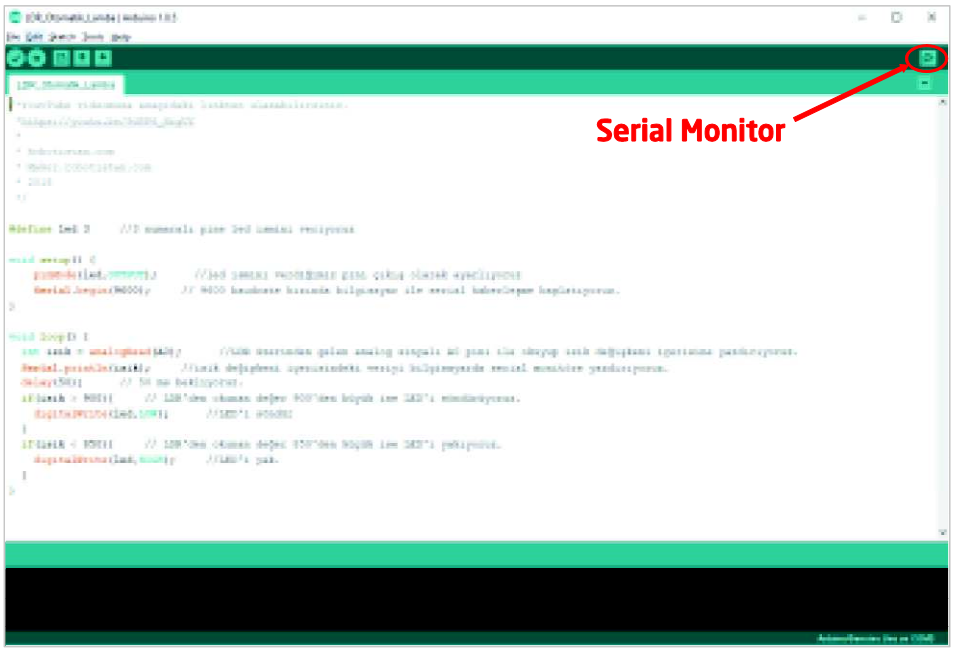


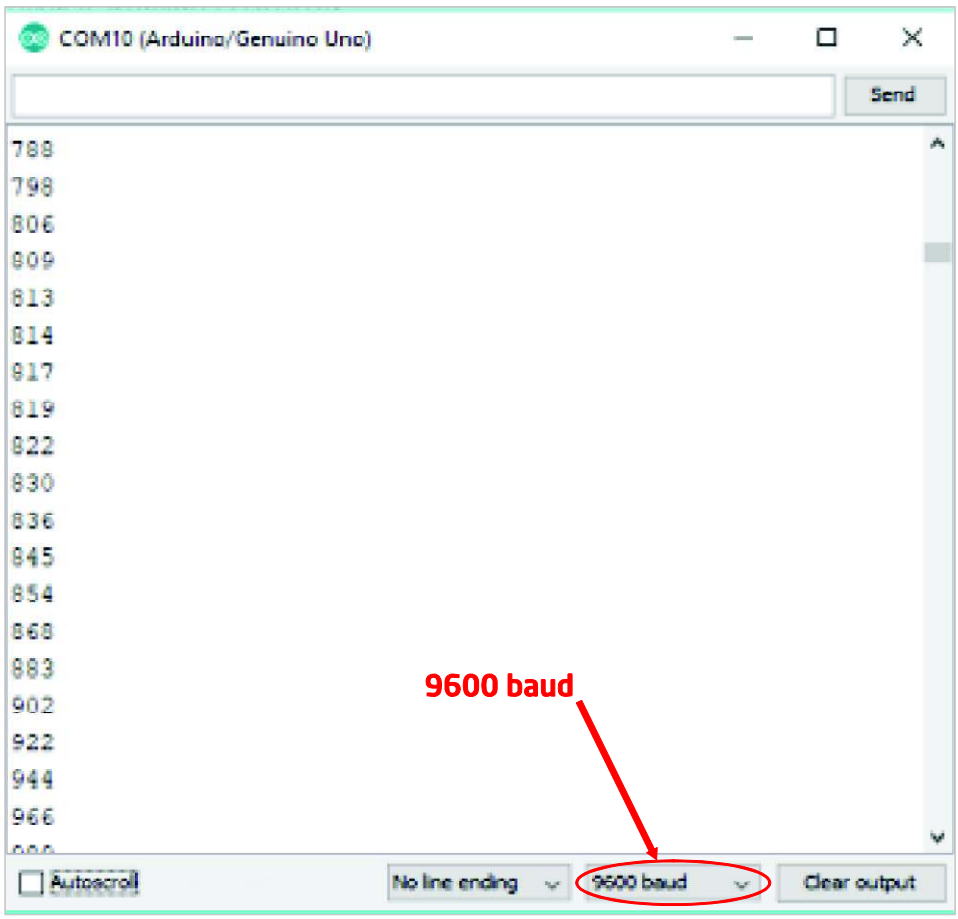
Kod kısmına geçecek olursak ilk satırımızda LED'i bağlayacağımız pine isim veriyoruz. Bu işlemi "#define" komutu ile yapacağız. Bu işlemden sonra artık ihtiyaç halinde 3 yazmak yerine "led" yazarak işlemleri kolaylaştıracağız.

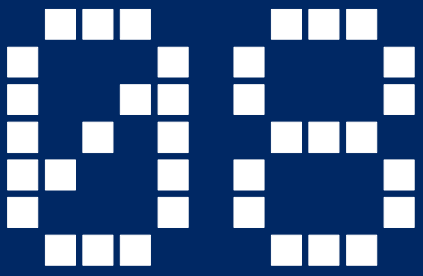
Kodun "setup" kısmında LED'imizi bağladığımız pini çıkış vermemiz ve seri haberleşmeyi başlatmamız gerekiyor. Seri haberleşmeyi 9600 baudrate hızında başlatıyoruz. Bu sayı bilgisayar ve Arduino kartının ne kadar hızlı haberleştiğini belirler. Bu sayıyı rasgele yazamıyoruz. Önceden belirlenmiş hızları kullanmamız gerekmektedir. 300,600,1200,2400,4800,9600,14400,19200,28800,38400 veya 115200 baudrate hızlarını kullanabilirsiniz. Arduino koduna yazdığınız baudrate değeri bilgisayarda açacağını seri monitör'ün sağ alt köşesindeki hız ile aynı olmalıdır.

Ana algoritmamızın döneceği "loop" içerisinde "int" tipinde ve "isik" isminde bir değişken tanımlayıp içerisinde LDR okuduğumu değeri yazdırıyoruz. Okuduğumuz bu değeri seri haberleşme üzerinden bilgisayara gönderiyoruz. 50 ms kadar bekledikten sonra "if" komutu ile gelen değerin istediğimiz değerlerin altında veya üstünde olup olmadığına göre değerlendirip karar veriyoruz. Ortamdaki ışık az ise LDR üzerinden gelen değer küçülecektir. Bizim ortamımızda 850 değerinden sonra LED'in yanmasını istiyoruz. Ortam aydınlanmaya başlayınca da değer artacağından dolayı 900 değerinden sonra sönmesini istiyoruz.

Kodu kartımıza attıktan sonra Arduino IDE'si içerisinde "Serial Monitor" butonuna tıklayıp gelen verileri görebiliriz. LDR üzerinde elinizi getirdiğinizde ışık şiddeti değişeceği için okudğunuz değerlerde değişecektir.







Arduino ile RGB LED Uygulaması

robotistan BLOG

Uygulamanın blog yazısına aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinodersleri

YouTube

Uygulamanın videosuna aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinovideodersler

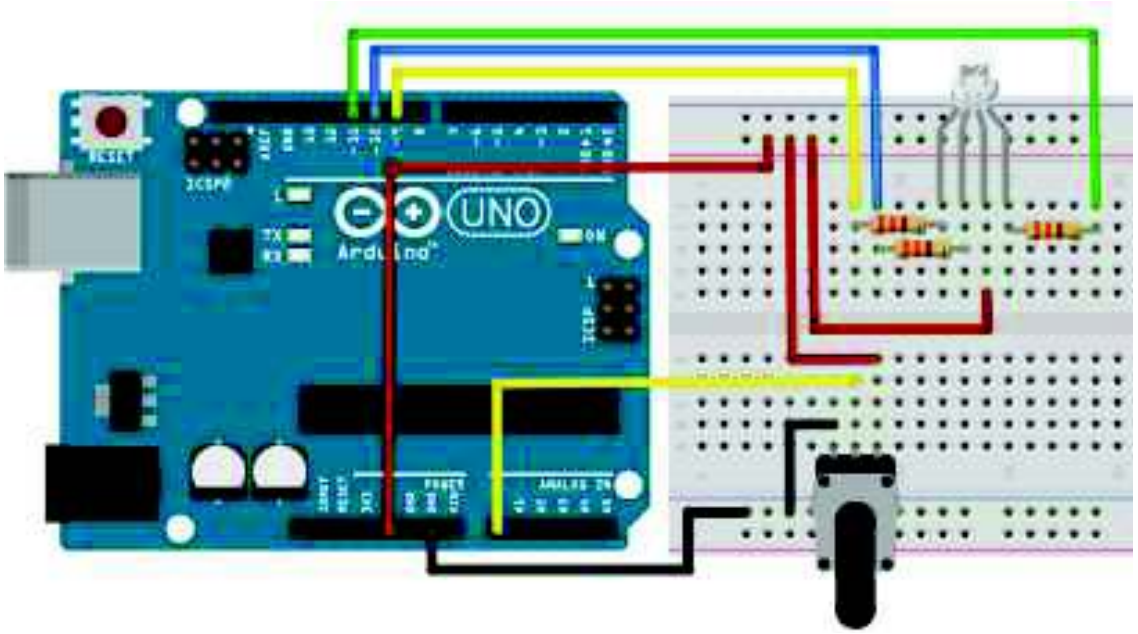
Gerekli malzemeler:

- Arduino Uno
- Breadboard
- 3 Adet 330 Ohm Direnç (Turuncu,Turuncu, Kahverengi)
- RGB LED
- 10K Potansiyometre
- 9 Adet Erkek-Erkek Jumper Kablo

RGB LED içerisinde kirmizi, yeşil ve mavi renkleri içeren 3 adet led yapısı bulunur. İçinde bulundurduğu renklerin baş harflerinin birleşmesi RGB (Red, Green, Blue) ismi oluşmuştur. 3 adet LED düşündüğümüzde her birinde bir artı bir eksi olacak şekilde toplam 6 bacak olması gerekir. Kullandığımız RGB LED'de 4 bacak bulunur. İçerideki 3 renk artı bacağını ortak olarak kullanır. Artı bacaktan enerji verildiğinde her renk ait olan eksi bacaktan eksiye bağlantı yapıldığında ilgili LED yanacaktır. RGB LED'lerin ortak artı yerine ortak eksi bacağı olanları da mevcut. Bu durumda ilgili LED'de artı sinyalini verdiğinizde yanacaktır. LED'leri tarif ederken ortak anot(artı) ve ortak katot(eksi) terimlerini kullanabilirsiniz. Bu durumda bizim kullanacağımız LED ortak anot olacaktır.

Bu uygulamamızda renkleri tam parlaklık dışında ara renklerde de yakmak istiyoruz. LED'i istediğimiz parlaklık seviyesinde yakabilmek için PWM özelliğini kullanmamız gerekiyor. PWM özelliği belirli bacaklarda (3,5,6,9,10,11) bulunduğu için bağlantılarımız yaparken bu bacakları kullanmaya dikkat edeceğiz. Hemen devremizi kurarak projemize başlayalım.

Bu uygulamamızda renkleri tam parlaklık dışında ara renklerde de yakmak istiyoruz. LED'i istediğimiz parlaklık seviyesinde yakabilmek için PWM özelliğini kullanmamız gerekiyor. PWM özelliği belirli bacaklarda (3,5,6,9,10,11) bulunduğu için bağlantılarımız yaparken bu bacakları kullanmaya dikkat edeceğiz. Hemen devremizi kurarak projemize başlayalım.



Kod kısmında diğer yazılımlarda yaptığımız gibi isim atamalarını yapıp değişkenlerimizi oluşturacağız. İsim ataması yaparken istersek burada yaptığımız gibi değişken ataması yapabilir. İnteger türünde "potPin" değişkeni oluşturup içerisine 3 yazabiliriz. Bu durumda "potPin" yazdığımız her yere 3 sayısı yazılmış gibi olacaktır. Bu tanımlamayı yaparken dilerseniz "define" komutunu da kullanabilirsiniz. Bu komutu kullanırken "#define potPin 3" şeklinde kullanılmalıdır. Eşittir ifadesi ve satır sonunda noktalı virgül koymaya gerek yoktur.

Arduino

```
int potPin = 3;      //
int potDeger = 0;    //

int kirmiziPin = 9;
int yesilPin = 10;
int maviPin = 11;


int kirmiziDeger = 0;
int yesilDeger = 0;
int maviDeger = 0;
```

```
void setup()
{
  pinMode(kirmiziPin, OUTPUT);
  pinMode(yesilPin, OUTPUT);
  pinMode(maviPin, OUTPUT);
}

void loop()
{
  potDeger = analogRead(potPin);

  if (potDeger < 341)
  {
    potDeger = (potDeger * 3) / 4;

    kirmiziDeger = 255 - potDeger;
    yesilDeger = potDeger;
    maviDeger = 0;
  }
  else if (potDeger < 682)
  {
    potDeger = ( (potDeger-341) * 3) / 4;

    kirmiziDeger = 0;
    yesilDeger = 255 - potDeger;
    maviDeger = potDeger;
  }
  else
  {
    potDeger = ( (potDeger-683) * 3) / 4;

    kirmiziDeger = potDeger;
    yesilDeger = 0;
    maviDeger = 255 - potDeger;
  }
  analogWrite(kirmiziPin, 255-kirmiziDeger);
  analogWrite(yesilPin, 255-yesilDeger);
  analogWrite(maviPin, 255-maviDeger);
```

Projenin "setup" kısmında ise çıkış vereceğimiz pinleri belirlememiz yeterli. Kırmızı, yeşil ve mavi LED'in eksi bacakları için 3 adet çıkış ihtiyacımız olacak.

Ana program döngüsüne geçtiğimizde ise "potPin" pininden okuduğumuz değeri değerlendirerek devam edeceğiz. Okumayı yaptıktan sonra hem "if", "ifelse" ve "else" komutlarını kullanarak karar verip gerekli LED'leri gelen değerlere göre yakacağız. İlk "if" komutunda gelen değer 341'den küçük ise "if" parantezi altındaki işlemleri yapmasını istiyoruz.

. "if" içerisinde "potDeger" içerisindeki değeri 0-255 arasında (PWM çıkış verebildiği değerler) oranlamak için 4'e bölüp, 3 ile çarpıyoruz. Bu sayede 340 değeri geldiğinde bu hesaplama sonucunda 255 değerini elde edeceğiz.

Analog pin üzerinden 0 ile 1023 arasında okuma yapabiliyoruz. Bu değeri 3 adet LED olduğu için 3 farklı bölgeye böldük. 0-1023 arasında bu bölgeler 0-341, 342-681, 682-1023 olarak belirledik. Gelen değerin bu bölgelerden hangisinde olduğunu belirlemek için "if-else" yapısını kullanıyoruz. Gelen değer "if" komutu ile değerlendirilir eğer istenen değerse "if" içerisi yapılır ve diğer koşullar (if else, else) atlanır. Eğer "if" koşulu sağlanamaz ise "if else" yani ikinci bölge değerlendirilir. Burası sağlanıp sağlanmadığına göre ya içerisindeki komutlar uygulanır yada "else" satırına geçilir. "if else" satırlarını çoğaltarak 3 basamak yerine birçok basamak belirleyebilirsiniz.

Her basamak içerisinde benzer komutlar uygulanıyor. Sadece atanan değer farklı renklerde oluyor. Örneğin "if" içerisini incelersek, gelen değer 0 ile 255 arasına oranlanıyor. Burada bir açıklama yapmamız gerekli bir LED'i PWM ile kontrol ederken 255 verdiğimizde tam parlaklıkta yanar. Ancak buradaki devrede LED'in artı bacağını değil eksi bacağını PWM pin'ine bağladığımız için ters ekti olacaktır. PWM çıkışından 0(sıfır) verdiğimizde LED tam parlaklıkta yanacak, 255 verdiğimizde sönecektir. Bu problemi ters durumu aşmak için çıkan değerleri LED'lere yollamadan önce 255'ten çıkararak yollayacağız. Bu durumda "if" koşulları içerisinde sanki normal LED bağlamış gibi kodumuzu yazabileceğiz. İlk "if" içerisinde kırmızı LED'e göndermek üzere 255'ten "potDeger"i çıkararak gönderiyoruz. Yeşil LED'e ise direk olarak potDeger'ini gönderiyoruz. Mavi LED'i söndürmek için 0(sıfır) değerini atıyoruz. Renkler arasında geçiş için 3 basamağın her birinde bir LED'i tamamen söndürüp diğer LED'lere gönderilen değerlerin toplamının 255 olmasını sağlıyoruz. Bu yöntem ile potansiyometreyi çevirdiğimizde bir rengin parlaklığı artarken diğeri azalıyor ve renk geçişleri meydana geliyor.

03

LED

220 OHM RESISTOR

TEMPERATURE SENSOR

INGREDIENTS

LOVE-O-METER

TURN THE ARDUINO INTO A LOVE MACHINE. USING AN ANALOG INPUT, YOU'RE GOING TO REGISTER JUST HOW HOT YOU REALLY ARE!

Discover: analog Input, using the serial monitor

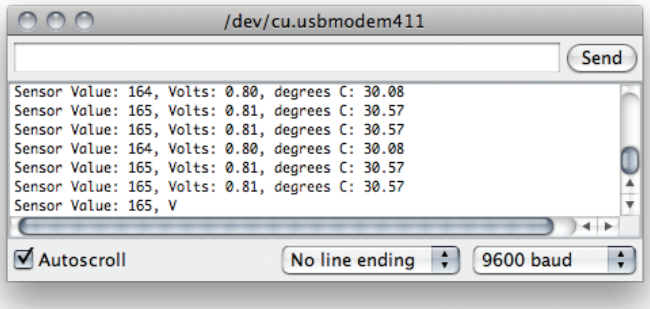
Time: **45 MINUTES**
Level: ■ □ □ □ □

Builds on projects: **1, 2**

While switches and buttons are great, there's a lot more to the physical world than on and off. Even though the Arduino is a digital tool, it's possible for it to get information from analog sensors to measure things like temperature or light. To do this, you'll take advantage of the Arduino's built-in Analog-to-Digital Converter (ADC). Analog in pins A0-A5 can report back a value between 0-1023, which maps to a range from 0 volts to 5 volts.

You'll be using a ***temperature sensor*** to measure how warm your skin is. This component outputs a changing voltage depending on the temperature it senses. It has three pins: one that connects to ground, another that connects to power, and a third that outputs a variable voltage to your Arduino. In the sketch for this project, you'll read the sensor's output and use it to turn LEDs on and off, indicating how warm you are. There are several different models of temperature sensor. This model, the TMP36, is convenient because it outputs a voltage that changes directly proportional to the temperature in degrees Celsius.

The Arduino IDE comes with a tool called the ***serial monitor*** that enables you to report back results from the microcontroller. Using the serial monitor, you can get information about the status of sensors, and get an idea about what is happening in your circuit and code as it runs.



Serial monitor

Fig. 1

BUILD THE CIRCUIT

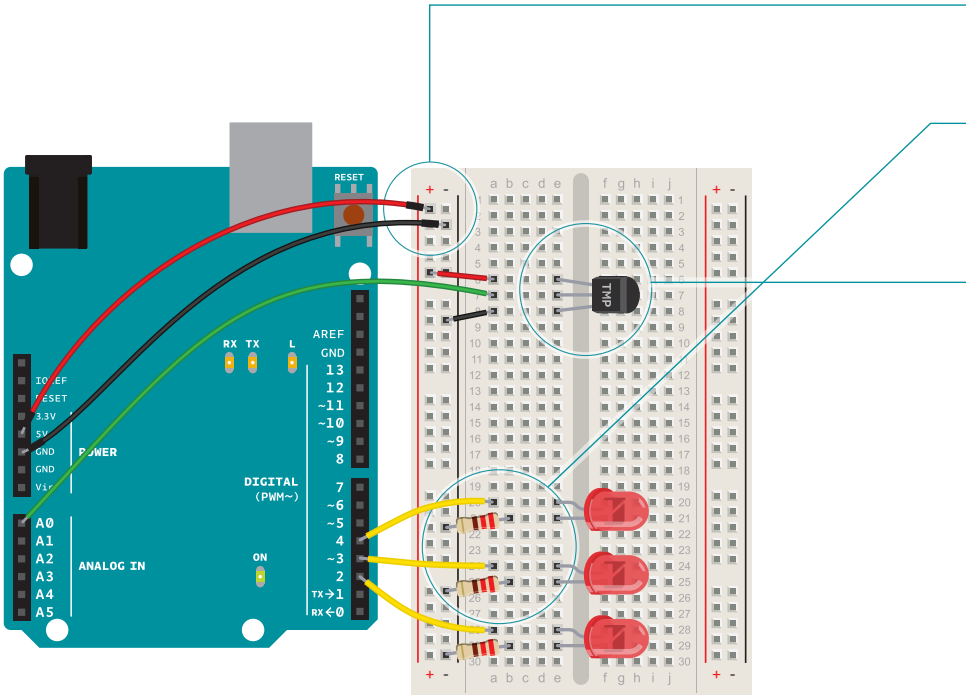


Fig. 2

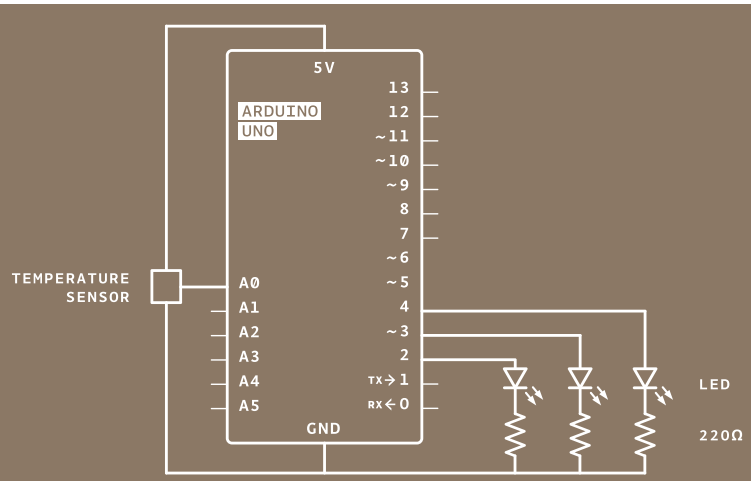


Fig. 3

In this project, you need to check the ambient temperature of the room before proceeding. You're checking things manually right now, but this can also be accomplished through calibration. It's possible to use a button to set the baseline temperature, or to have the Arduino take a sample before starting the `loop()` and use that as the reference point. Project 6 gets into details about this, or you can look at the Calibration example that comes bundled with the Arduino software:

arduino.cc/calibration

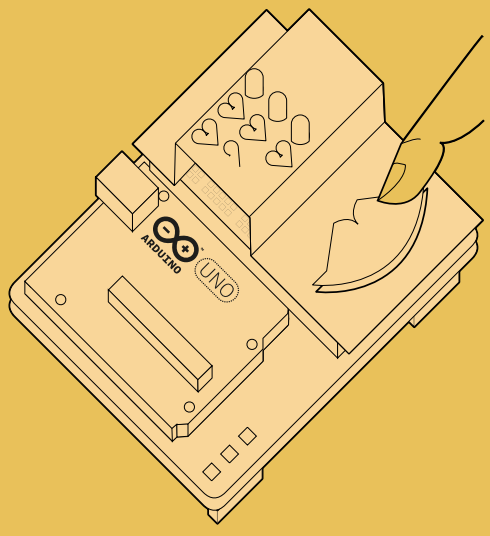
1. Just as you've been doing in the earlier projects, wire up your breadboard so you have power and ground.

2. Attach the cathode (short leg) of each of the LEDs you're using to ground through a 220-ohm resistor. Connect the anodes of the LEDs to pins 2 through 4. These will be the indicators for the project.

3. Place the TMP36 on the breadboard with the rounded part facing away from the Arduino (the order of the pins is important!) as shown in Fig. 2. Connect the left pin of the flat facing side to power, and the right pin to ground. Connect the center pin to pin A0 on your Arduino. This is analog input pin 0.

Create an interface for your sensor for people interact with. A paper cutout in the shape of a hand is a good indicator. If you're feeling lucky, create a set of lips for someone to kiss, see how well that lights things up! You might also want to label the LEDs to give them some meaning. Maybe one LED means you're a cold fish, two LEDs means you're warm and friendly, and three LEDs means you're too hot to handle!

1. Cut out a piece of paper that will fit over the breadboard. Draw a set of lips where the sensor will be, and cut some circles for the LEDs to pass through.



2. Place the cutout over the breadboard so that the lips cover the sensor and the LEDs fit into the holes. Press the lips to see how hot you are!

THE CODE

A pair of useful constants

Constants are similar to variables in that they allow you to uniquely name things in the program, but unlike variables they cannot change. Name the analog input for easy reference, and create another named constant to hold the baseline temperature. For every 2 degrees above this baseline, an LED will turn on.
You've already seen the int datatype, used here to identify which pin the sensor is on. The temperature is being stored as a *float*, or floating-point number. This type of number has a decimal point, and is used for numbers that can be expressed as fractions.

Initialize the serial port to the desired speed

In the setup you're going to use a new command, `Serial.begin()`. This opens up a connection between the Arduino and the computer, so you can see the values from the analog input on your computer screen.
The argument `9600` is the speed at which the Arduino will communicate, 9600 bits per second. You will use the Arduino IDE's serial monitor to view the information you choose to send from your microcontroller. When you open the IDE's serial monitor verify that the baud rate is 9600.

Initialize the digital pin directions and turn off

Next up is a `for()` loop to set some pins as outputs. These are the pins that you attached LEDs to earlier. Instead of giving them unique names and typing out the `pinMode()` function for each one, you can use a `for()` loop to go through them all quickly. This is a handy trick if you have a large number of similar things you wish to iterate through in a program. Tell the `for()` loop to run through pins 2 to 4 sequentially.

Read the temperature sensor

In the `loop()`, you'll use a local variable named `sensorVal` to store the reading from your sensor. To get the value from the sensor, you call `analogRead()` that takes one argument: what pin it should take a voltage reading on. The value, which is between 0 and 1023, is a representation of the voltage on the pin.

Send the temperature sensor values to the computer

The function `Serial.print()` sends information from the Arduino to a connected computer. You can see this information in your serial monitor. If you give `Serial.print()` an argument in quotation marks, it will print out the text you typed. If you give it a variable as an argument, it will print out the value of that variable.

```
1 const int sensorPin = A0;
2 const float baselineTemp = 20.0;
```

```
3 void setup(){
4   Serial.begin(9600); // open a serial port
```

```
5   for(int pinNumber = 2; pinNumber<5; pinNumber++){
6     pinMode(pinNumber,OUTPUT);
7     digitalWrite(pinNumber, LOW);
8   }
9 }
```

for() loop tutorial
arduino.cc/for

```
10 void loop(){
11   int sensorVal = analogRead(sensorPin);
```

```
12   Serial.print("Sensor Value: ");
13   Serial.print(sensorVal);
```

Convert sensor reading to voltage

With a little math, it's possible to figure out what the real voltage on the pin is. The voltage will be a value between 0 and 5 volts, and it will have a fractional part (for example, it might be 2.5 volts), so you'll need to store it inside a `float`. Create a variable named voltage to hold this number. Divide `sensorVal` by 1024.0 and multiply by 5.0. The new number represents the voltage on the pin.

Just like with the sensor value, you'll print this out to the serial monitor.

Convert the voltage to temperature and send the value to the computer

If you examine the sensor's *datasheet*, there is information about the range of the output voltage. Datasheets are like manuals for electronic components. They are written by engineers, for other engineers. The datasheet for this sensor explains that every 10 millivolts of change from the sensor is equivalent to a temperature change of 1 degree Celsius. It also indicates that the sensor can read temperatures below 0 degrees. Because of this, you'll need to create an offset for values below freezing (0 degrees). If you take the voltage, subtract 0.5, and multiply by 100, you get the accurate temperature in degrees Celsius. Store this new number in a floating point variable called temperature.

Now that you have the real temperature, print that out to the serial monitor too. Since the temperature variable is the last thing you're going to be printing out in this loop, you're going to use a slightly different command: `Serial.println()`. This command will create a new line in the serial monitor after it sends the value. This helps make things easier to read in when they are being printed out.

Turn off LEDs for a low temperature

With the real temperature, you can set up an `if()...else` statement to light the LEDs. Using the baseline temperature as a starting point, you'll turn on one LED on for every 2 degrees of temperature increase above that baseline. You're going to be looking for a range of values as you move through the temperature scale.

```
14   // convert the ADC reading to voltage
15   float voltage = (sensorVal/1024.0) * 5.0;
```

```
16   Serial.print(", Volts: ");
17   Serial.print(voltage);
```

```
18   Serial.print(", degrees C: ");
19   // convert the voltage to temperature in degrees
20   float temperature = (voltage - .5) * 100;
21   Serial.println(temperature);
```

```
22   if(temperature < baselineTemp){
23     digitalWrite(2, LOW);
24     digitalWrite(3, LOW);
25     digitalWrite(4, LOW);
```

Starter Kit datasheets
arduino.cc/kitdatasheets

Turn on one LED for a low temperature

The && operator means "`and`", in a logical sense. You can check for multiple conditions: "if the temperature is 2 degrees greater than the baseline, and it is less than 4 degrees above the baseline."

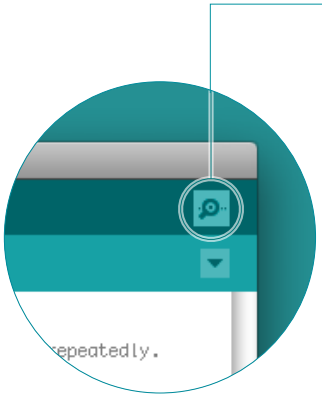
Turn on two LEDs for a medium temperature

If the temperature is between two and four degrees above the baseline, this block of code turns on the LED on pin 3 as well.

Turn on three LEDs for a high temperature

The Analog-to-Digital Converter can only read so fast, so you should put a small delay at the very end of your `loop()`. If you read from it too frequently, your values will appear erratic.

USE IT



With the code uploaded to the Arduino, click the serial monitor icon. You should see a stream of values coming out, formatted like this : `Sensor: 200, Volts: .70, degrees C: 17`

Try putting your fingers around the sensor while it is plugged into the breadboard and see what happens to the values in the serial monitor. Make a note of what the temperature is when the sensor is left in the open air.

Close the serial monitor and change the baselineTemp constant in your program to the value you observed the temperature to be. Upload your code again, and try holding the sensor in your fingers. As the temperature rises, you should see the LEDs turn on one by one. Congratulations, hot stuff!

```
  }else if(temperature >= baselineTemp+2 &&
    temperature < baselineTemp+4){
    digitalWrite(2, HIGH);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);

  }else if(temperature >= baselineTemp+4 &&
    temperature < baselineTemp+6){
    digitalWrite(2, HIGH);
    digitalWrite(3, HIGH);
    digitalWrite(4, LOW);

  }else if(temperature >= baselineTemp+6){
    digitalWrite(2, HIGH);
    digitalWrite(3, HIGH);
    digitalWrite(4, HIGH);

  }
  delay(1);
}
```

Create an interface for two people to test their compatibility with each other. You get to decide what compatibility means, and how you'll sense it. Perhaps they have to hold hands and generate heat? Maybe they have to hug? What do you think?

Expanding the types of inputs you can read, you've used analogRead() and the serial monitor to track changes inside your Arduino. Now it's possible to read a large number of analog sensors and inputs.

10

Ultrasonik Sensör ile Park Sensörü Yapımı

robotistan BLOG

Uygulamanın blog yazısına aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinodersleri

YouTube

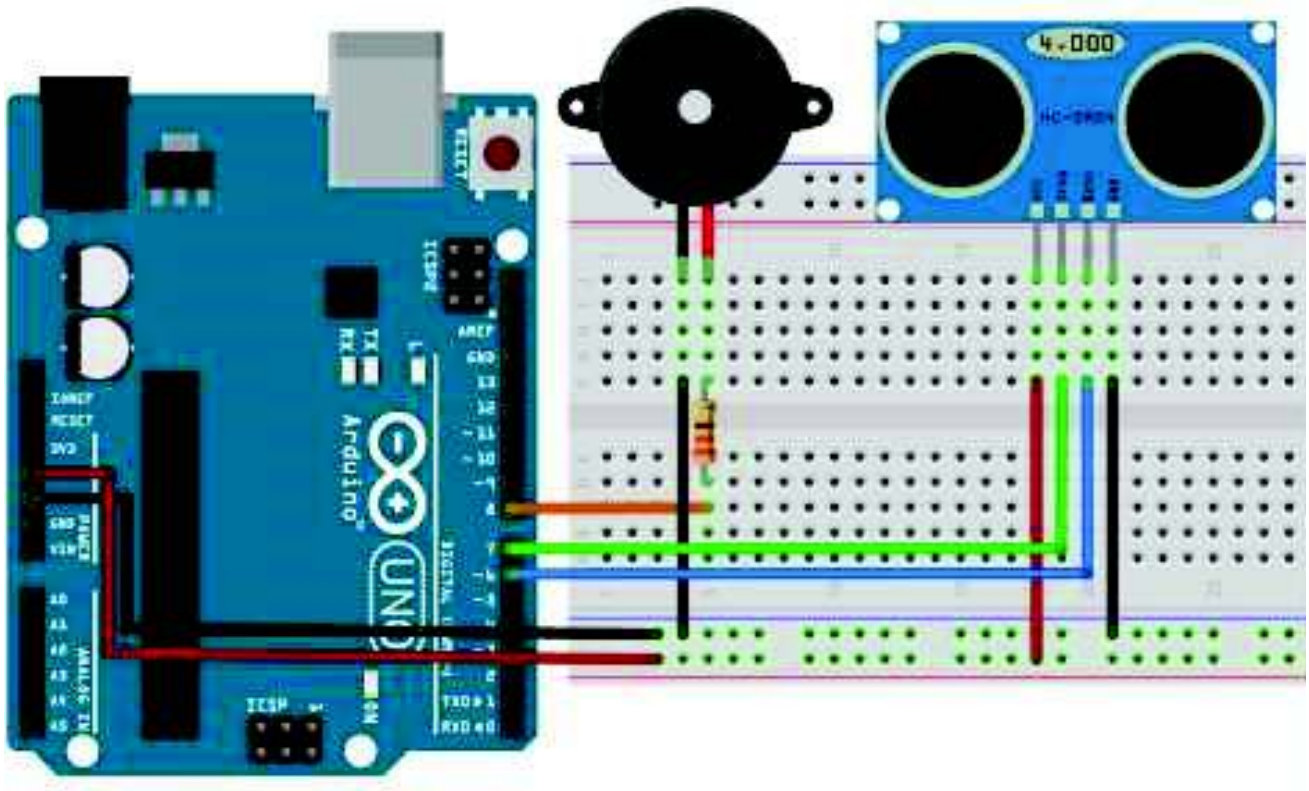
Uygulamanın videosuna aşağıdaki linkten ulaşabilirsiniz.
http://bit.ly/arduinovideodersler

Gerekli malzemeler:

- Arduino Uno
- Breadboard
- 8 Adet Erkek-Erkek Jumper Kablo
- Buzzer
- 330 Ohm Direnç (Turuncu-Turuncu-Kahverengi)
- HC-SR04 Ultrasonik Sensör

Ultrasonik sensör uygulamamızda yeni bir kullanım örneği göreceğiz. HC-SR04 ultrasonik sensörü üzerinde bir tane ses gönderebilen, bir tane de ses algılayabilen metal kısımlar bulunuyor. Sensörden ses gönderildikten sonra eğer önünde cisim veya engelden yansıyarak tekrar sensöre geliyor. Alıcı kısım yansıyan sinyali algılayarak ölçüm yapabiliyor.

Mesafeyi ölçmek için sinyal gönderdikten itibaren geri gelene kadarki süreyi ölçüyoruz. Sesin havadaki hızını bildiğimiz için süre ve hızdan toplam mesafeyi hesaplayabiliyoruz. Hemen devremizi kurarak devam edelim.



Arduino

```
#define echoPin 6
#define trigPin 7
#define buzzerPin 8

int maximumRange = 50;
int minimumRange = 0;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(buzzerPin, OUTPUT);
}

void loop() {
  int olcum = mesafe(maximumRange, minimumRange);
  melodi(olcum*10);
}

int mesafe(int maxrange, int minrange)
{
  long duration, distance;

  digitalWrite(trigPin,LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = duration / 58.2;
  delay(50);     //50 ms bekliyoruz.

  if(distance >= maxrange || distance <= minrange)
  return 0;
  return distance;
}

int melodi(int dly)
{
  tone(buzzerPin, 440);
  delay(dly);
  noTone(buzzerPin);
  delay(dly);
}
```

Yazılım kısmında"#define" komutları ile kullanacağımız pinlere isimler veriyoruz. "maximumRange" ve "minimumRange" isminde "integer"(tamsayı) tipinde değişkenler tanımlıyoruz. "setup" kısmında giriş ve çıkış olacak pinleri ayarlıyoruz.

Ana program döngümüz çok kısa görünüyor. Bu kısımda ilk olarak mesafe fonksiyonuna gidiyoruz."long" türünde "duration" ve "distance" değişkenleri tanımlanıyor."long" Önceden kullandığımız "integer" gibi bir değişken. İçerisinde "integer" değişkenine göre çok daha büyük sayılar tutabilir. +2,147,483,647'den, -2,147,483,647'ye kadar içerisine atanabilmektedir. tutabilmektedir. Tutabildiği sayı hacminden dolayı tanımlandığında "integer" değişkenine göre 2 kat fazla hafıza kullanır. Değişken tanımladıktan sonra sensörün "trig" pinini yüksek ve alçak yaparak sensörün fiziksel ortama ses dalgası yollamasını sağlıyoruz. Ses dalgası yollandıktan sonra "pulseIn(echoPin,HIGH)" komutu ile yolladığımız ses dalgasının cisimden yansıyıp geri gelmesini bekliyoruz. Bu beklediğimiz zamanıda "pulseIn" komutu ile ölçebiliyoruz. Ölçtüğümüz bu değer "duration" değişkenine yazdırıyoruz. Süre ölçüldüğüne göre şimdi mesafeyi hesaplamaya geldi. Ölçtüğümüz süreyi sesin hızına göre mesafeye çevirmek için "58.2"ye bölüyoruz. Mesafe değerine ulaşınca, bu değer sensörün ölçebildiği minimum (2 cm) ve maksimum (400 cm) arasında değilse 0(sıfır) değeri ile dönüş yap diyoruz. İstediğimiz araralıkta ise tekrar ana fonksiyona dönerek "olcum" değişkeni içerisine veri yazılıyor.

Ana fonksiyonumuzda "melodi" fonksiyonuna olcum değişkeninin içindeki değer 10 ile çarpılıp gönderiliyor. Bu değer "melodi" içerisindeki bekleme sürelerinde kullanılarak 2 dıt sesi arasındaki süreyi belirleyecek. Eğer sensör az mesafe ölçüyor ise kısa aralıklar ile, eğer sensör uzun mesafe algılıyor ise uzun aralıklar ile ötecek.